CS634

DATA MINING PROJECT

vm567

TITLE : United States Patent and Trademark Office

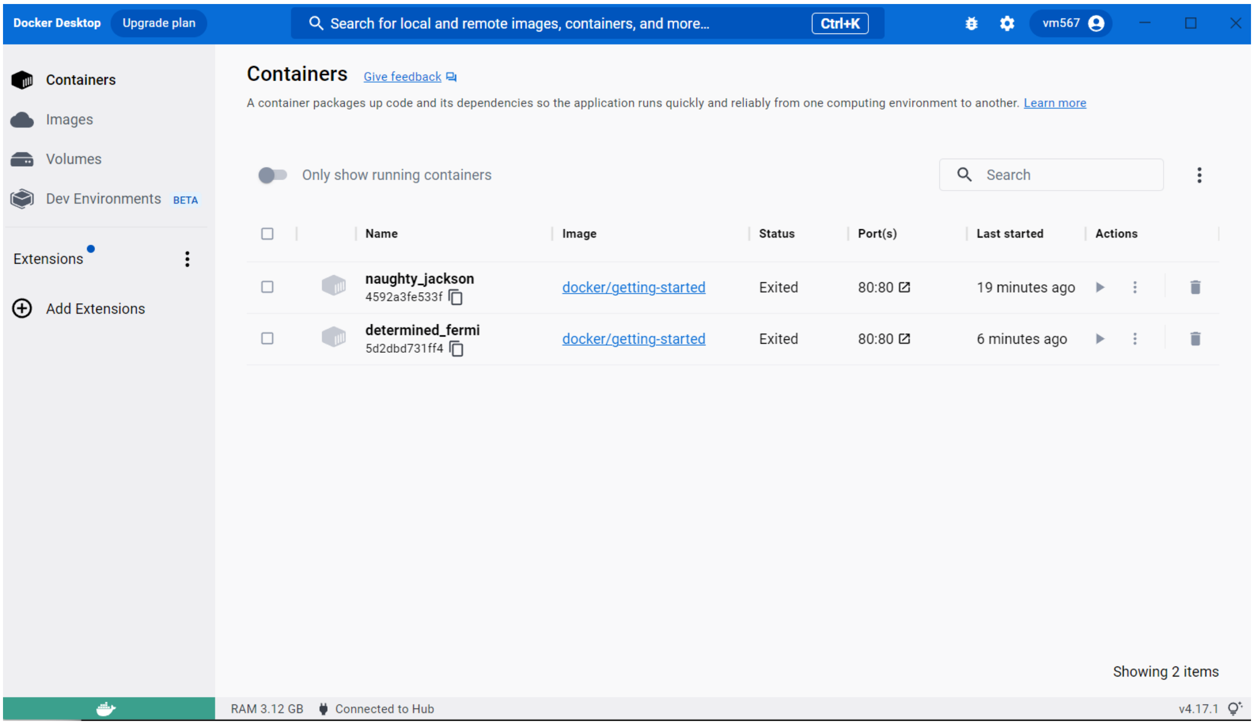
Abstract: A federal organization called the United States Patent and Trademark Office (USPTO) is in charge of issuing patents and registering trademarks in the country. The USPTO, which reports to the Department of Commerce, is a key player in promoting innovation and defending intellectual property rights. The USPTO's roles, responsibilities, and importance in promoting and protecting intellectual property are briefly described in this abstract.

Milestone 1

Goal: To learn the basics of the Docker and create a development environment.

Steps:

1. Firstly, Install dockers application(latest) from google.



Containers Give feedback

A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. Learn more

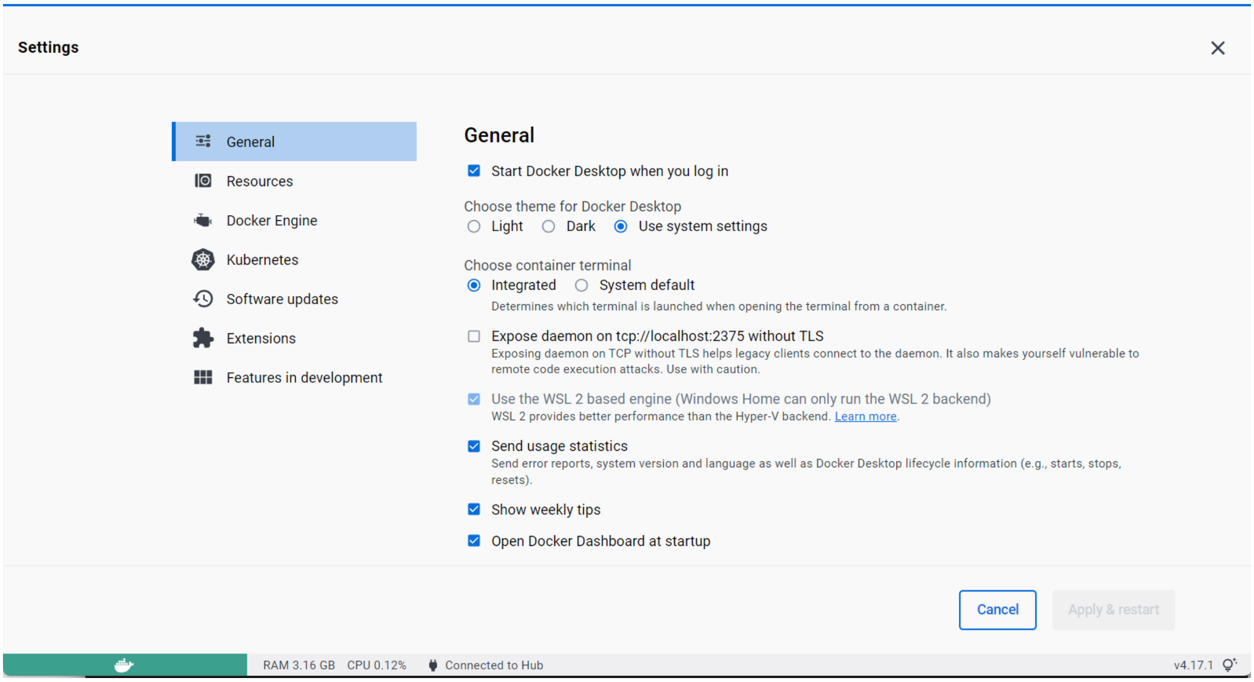
□	Name	Image	Status	Port(s)	Last started	Actions
□	naughty_jackson 4592a3fe533f	docker/getting-started	Exited	80:80	19 minutes ago	
□	determined_fermi 5d2dbd731ff4	docker/getting-started	Exited	80:80	6 minutes ago	

2. Go to settings, select general and check for the WSL which was selected by default in my system

3. Update the WSL to the latest version.

"https://code.visualstudio.com/docs/remote/wsl-tutorial" and

" https://docs.docker.com/desktop/windows/wsl/"



Settings

General

☑ Start Docker Desktop when you log in

Choose theme for Docker Desktop
○ Light ○ Dark ● Use system settings

Choose container terminal
● Integrated ○ System default
Determines which terminal is launched when opening the terminal from a container.

□ Expose daemon on tcp://localhost:2375 without TLS
Exposing daemon on TCP without TLS helps legacy clients connect to the daemon. It also makes yourself vulnerable to remote code execution attacks. Use with caution.

☑ Use the WSL 2 based engine (Windows Home can only run the WSL 2 backend)
WSL 2 provides better performance than the Hyper-V backend. Learn more.

☑ Send usage statistics
Send error reports, system version and language as well as Docker Desktop lifecycle information (e.g., starts, stops, resets).

☑ Show weekly tips

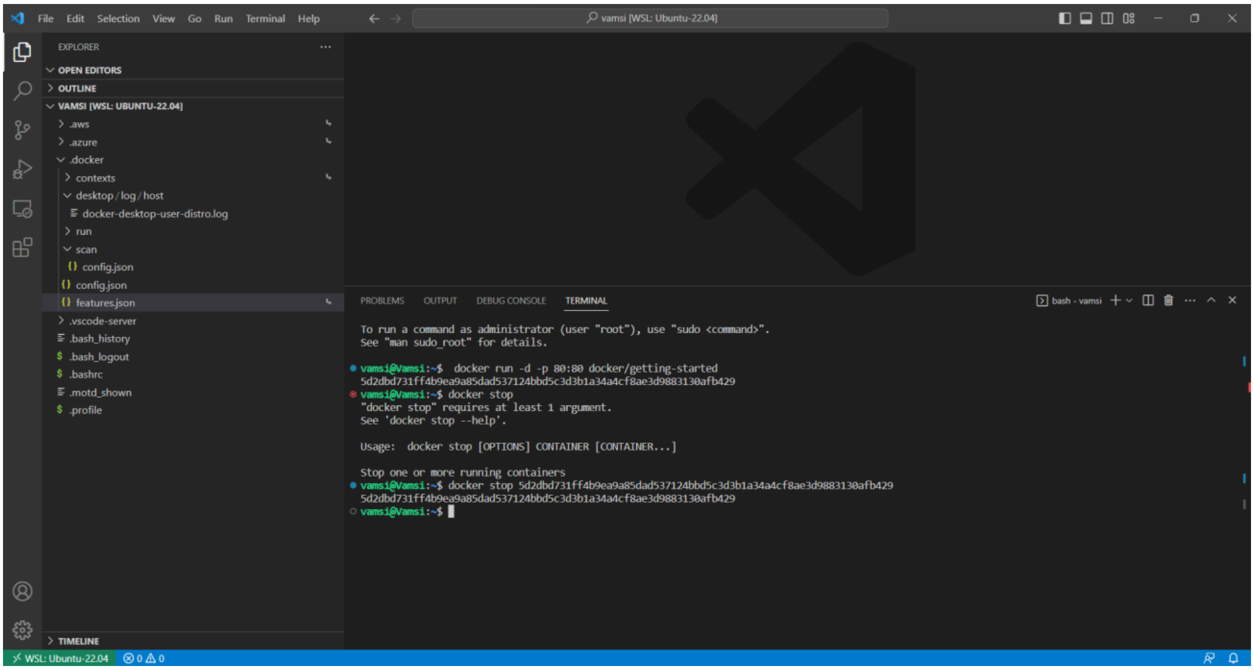
☑ Open Docker Dashboard at startup

RAM 3.16 GB CPU 0.12% Connected to Hub v4.17.1

4. Now, install Ubuntu

5. Open the Visual Studio code and download the WSL extension pack and then install remote control development extension pack which sets up a development environment for the system. Click ctrl+shift+p and connect to "WSL : connect to the default distro"

6. Now, open terminal and write "code ."



```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

vamsi@Vamsi:~$  docker run -d -p 80:80 docker/getting-started
5d2dbd731ff4b9ea9a85dad537124bbd5c3d3b1a34a4cf8ae3d9883130afb429
vamsi@Vamsi:~$ docker stop
"docker stop" requires at least 1 argument.
See 'docker stop --help'.

Usage:  docker stop [OPTIONS] CONTAINER [CONTAINER...]

Stop one or more running containers
vamsi@Vamsi:~$ docker stop 5d2dbd731ff4b9ea9a85dad537124bbd5c3d3b1a34a4cf8ae3d9883130afb429
5d2dbd731ff4b9ea9a85dad537124bbd5c3d3b1a34a4cf8ae3d9883130afb429
vamsi@Vamsi:~$ 
```

Milestone 2

Goal: Create a streamlit app and deploy a sentiment analysis app in hugging face model.

Steps:

1) created a virtual env to install necessary libraries (in vscode use command ctrl+shift+p and enter python create environment) vscode automatically creates a virtual environment for us
2) install necessary libraries
3) write all the libraries in a file (cmnd pip freeze > requirements.txt)

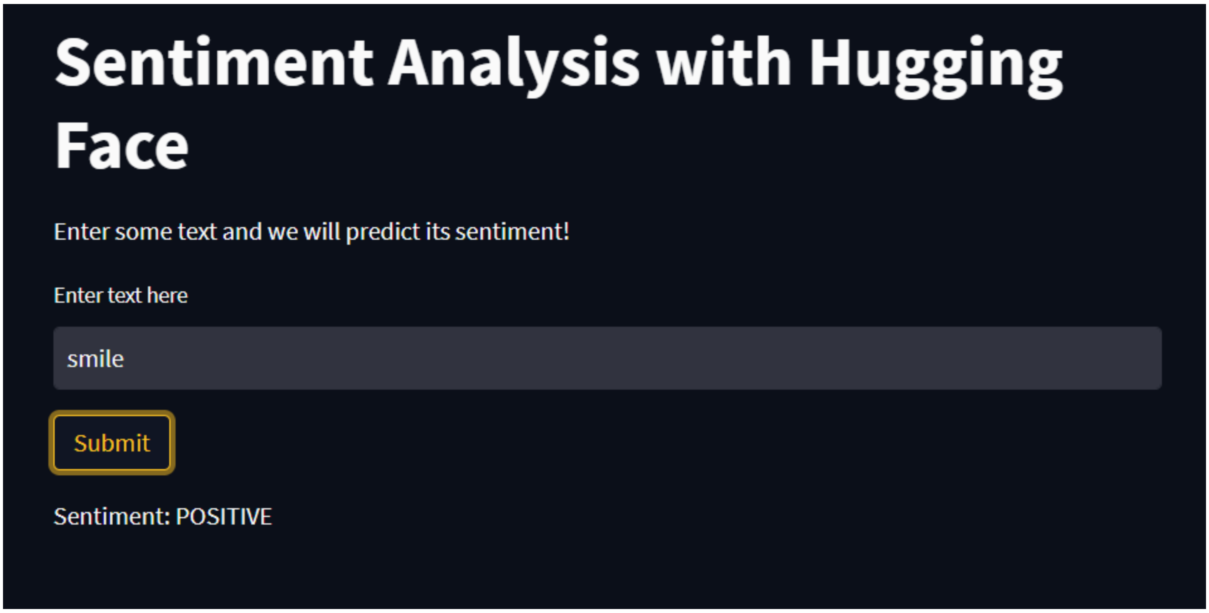
4) create a file name app.py

5) push the code to your feature branch

6) follow this link for more reference https://huggingface.co/docs/hub/spaces-github-actions

7) URL to the deployed application----- https://huggingface.co/spaces/vm567/sentiment-analysis-app

8) The output:



Sentiment Analysis with Hugging Face

Enter some text and we will predict its sentiment!

Enter text here

smile

Sentiment: POSITIVE

Milestone 3

Goal: To develop a classifier that analyses the given data and predicts the patentability score to determine how likely is the data going to get the patent right.

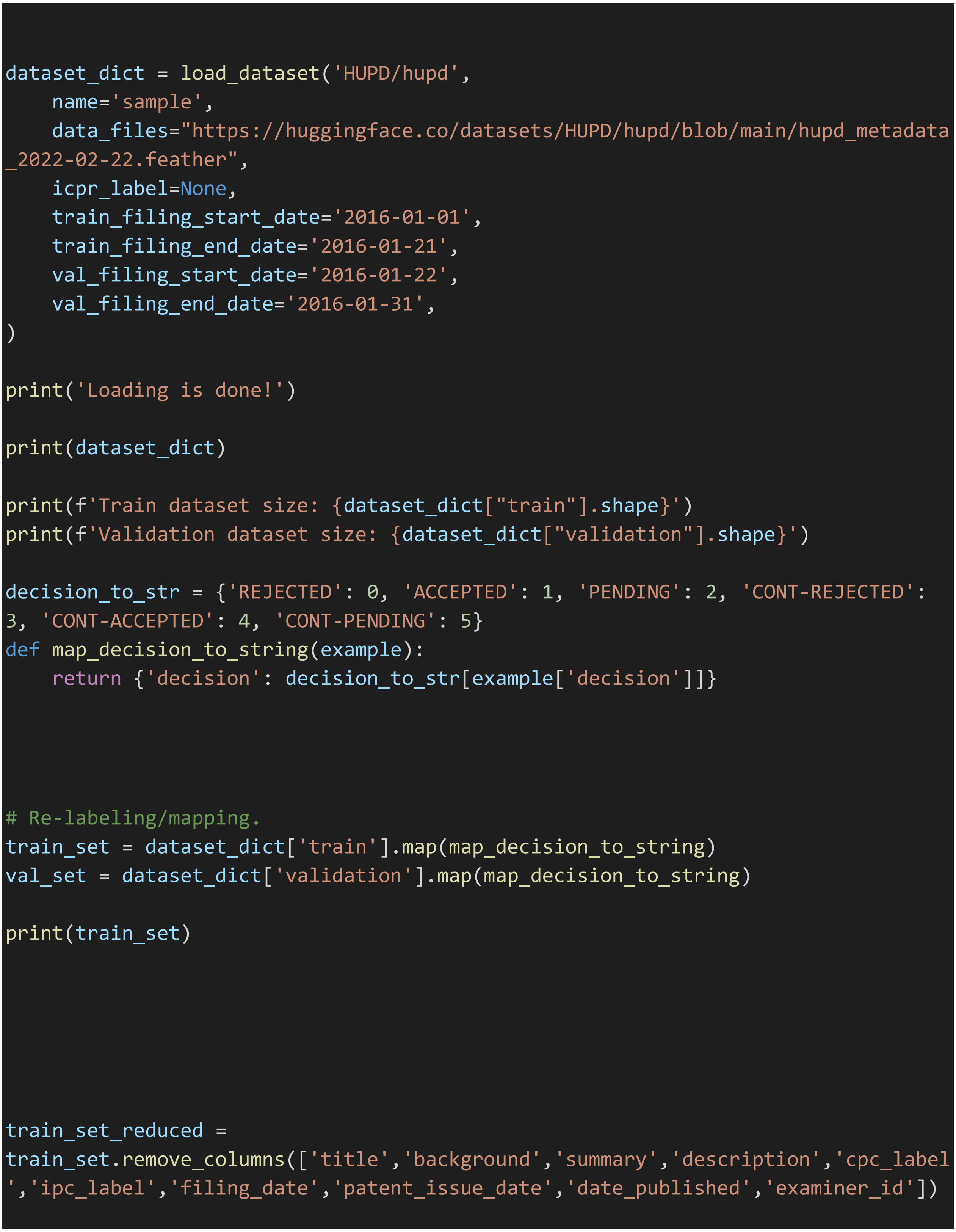
Steps:

United States Patent and Trademark Office application predicts a patentability score of a dataset based on the information provided in the dataset.

You can get the hugging face application here : https://huggingface.co/spaces/vm567/Finetuning_HUPD_dataset

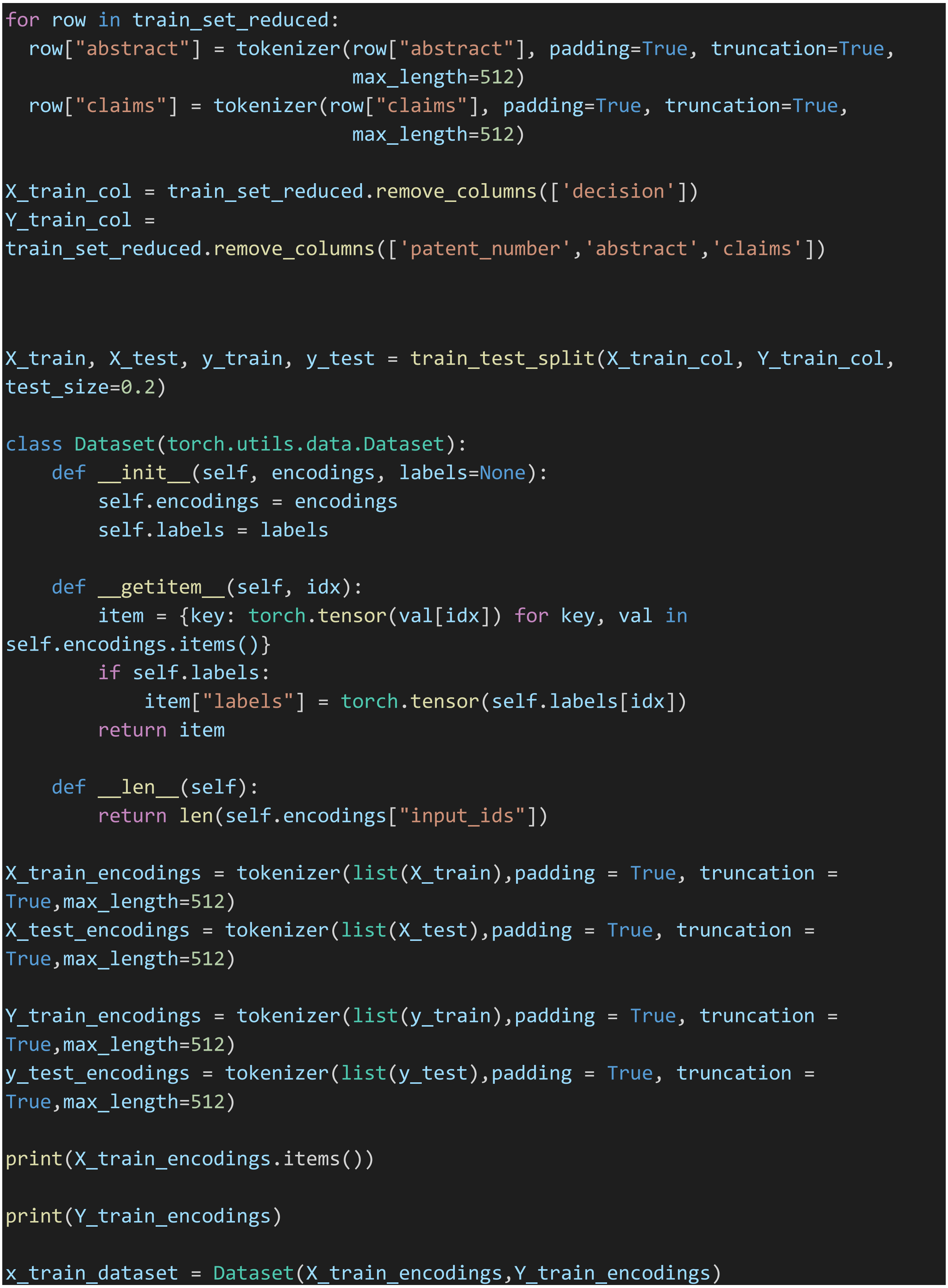
Source code :

```
from pprint import pprint

from datasets import load_dataset

from transformers import AutoTokenizer,pipeline

from torch.utils.data import DataLoader

import streamlit as st


import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, recall_score, precision_score,
f1_score
import torch
from transformers import TrainingArguments, Trainer
from transformers import BertTokenizer,
BertForSequenceClassification,AutoTokenizer,AutoModelForSequenceClassification
from transformers import DistilBertForSequenceClassification,
DistilBertTokenizer, DistilBertConfig
```



```
dataset_dict = load_dataset('HUPD/hupd',
    name='sample',
    data_files="https://huggingface.co/datasets/HUPD/hupd/blob/main/hupd_metadata
_2022-02-22.feather",
    icpr_label=None,
    train_filing_start_date='2016-01-01',
    train_filing_end_date='2016-01-21',
    val_filing_start_date='2016-01-22',
    val_filing_end_date='2016-01-31',
)

print('Loading is done!')

print(dataset_dict)

print(f'Train dataset size: {dataset_dict["train"].shape}')
print(f'Validation dataset size: {dataset_dict["validation"].shape}')

decision_to_str = {'REJECTED': 0, 'ACCEPTED': 1, 'PENDING': 2, 'CONT-REJECTED':
3, 'CONT-ACCEPTED': 4, 'CONT-PENDING': 5}
def map_decision_to_string(example):
    return {'decision': decision_to_str[example['decision']]}




# Re-labeling/mapping.
train_set = dataset_dict['train'].map(map_decision_to_string)
val_set = dataset_dict['validation'].map(map_decision_to_string)

print(train_set)






train_set_reduced =
train_set.remove_columns(['title','background','summary','description','cpc_label
','ipc_label','filing_date','patent_issue_date','date_published','examiner_id'])
```

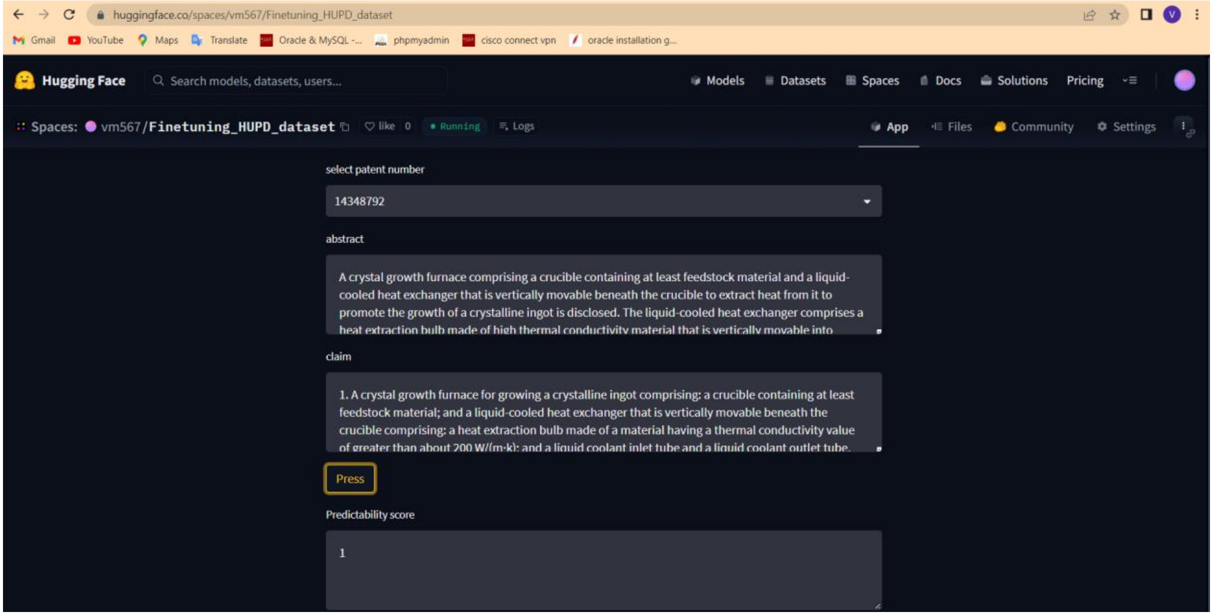
```
val_set_reduced =
val_set.remove_columns(['title','background','summary','description','cpc_label',
'ipc_label','filing_date','patent_issue_date','date_published','examiner_id'])

print(train_set_reduced)

train_set_reduced = train_set_reduced.filter(lambda row: row["decision"] < 2)
val_set_reduced = val_set_reduced.filter(lambda row: row["decision"] < 2)


print(train_set_reduced['decision'])
print(type(train_set_reduced))


# for the app

train_df_app=train_set_reduced.data.to_pandas()
val_set_app =val_set_reduced.data.to_pandas()


option = st.selectbox('select patent number',train_df_app['patent_number'])
idx_pos = list(np.where(train_df_app['patent_number'] == option))

abstract_text = train_df_app['abstract'].iloc[idx_pos[0][0]]
claim_text = train_df_app['claims'].iloc[idx_pos[0][0]]
decision_text = train_df_app['decision'].iloc[idx_pos[0][0]]


st.text_area("abstract",abstract_text)
st.text_area("claim",claim_text)


if st.button("Press"):
    st.text_area("Predictability score",decision_text)


# --------------------------------





tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertForSequenceClassification.from_pretrained('bert-base-uncased',num_labels=2)
```



```
for row in train_set_reduced:
  row["abstract"] = tokenizer(row["abstract"], padding=True, truncation=True,
                              max_length=512)
  row["claims"] = tokenizer(row["claims"], padding=True, truncation=True,
                              max_length=512)

X_train_col = train_set_reduced.remove_columns(['decision'])
Y_train_col =
train_set_reduced.remove_columns(['patent_number','abstract','claims'])



X_train, X_test, y_train, y_test = train_test_split(X_train_col, Y_train_col,
test_size=0.2)

class Dataset(torch.utils.data.Dataset):
    def __init__(self, encodings, labels=None):
        self.encodings = encodings
        self.labels = labels

    def __getitem__(self, idx):
        item = {key: torch.tensor(val[idx]) for key, val in
self.encodings.items()}
        if self.labels:
            item["labels"] = torch.tensor(self.labels[idx])
        return item

    def __len__(self):
        return len(self.encodings["input_ids"])

X_train_encodings = tokenizer(list(X_train),padding = True, truncation =
True,max_length=512)
X_test_encodings = tokenizer(list(X_test),padding = True, truncation =
True,max_length=512)

Y_train_encodings = tokenizer(list(y_train),padding = True, truncation =
True,max_length=512)
y_test_encodings = tokenizer(list(y_test),padding = True, truncation =
True,max_length=512)

print(X_train_encodings.items())

print(Y_train_encodings)

x_train_dataset = Dataset(X_train_encodings,Y_train_encodings)
```

```
X_test_dataset = Dataset(X_test_encodings,y_test_encodings)

print(x_train_dataset)

print(type(X_train_encodings))
print(type(x_train_dataset))

def compute_metrics(p):
    print(type(p))
    pred, labels = p
    pred = np.argmax(pred, axis=1)

    accuracy = accuracy_score(y_true=labels, y_pred=pred)
    recall = recall_score(y_true=labels, y_pred=pred)
    precision = precision_score(y_true=labels, y_pred=pred)
    f1 = f1_score(y_true=labels, y_pred=pred)

    return {"accuracy": accuracy, "precision": precision, "recall": recall, "f1":
f1}

# Define Trainer
args = TrainingArguments(
    output_dir="output",
    num_train_epochs=1,
    per_device_train_batch_size=8

)
trainer = Trainer(
    model=model,
    args=args,
    train_dataset=x_train_dataset,
    eval_dataset=X_test_dataset,
    compute_metrics=compute_metrics
)

# trainer.train()
```

OUTPUT:



select patent number

14348792

abstract

A crystal growth furnace comprising a crucible containing at least feedstock material and a liquid-cooled heat exchanger that is vertically movable beneath the crucible to extract heat from it to promote the growth of a crystalline ingot is disclosed. The liquid-cooled heat exchanger comprises a heat extraction bulb made of high thermal conductivity material that is vertically movable into

claim

1. A crystal growth furnace for growing a crystalline ingot comprising: a crucible containing at least feedstock material; and a liquid-cooled heat exchanger that is vertically movable beneath the crucible comprising: a heat extraction bulb made of a material having a thermal conductivity value of greater than about 200 W/(m·k); and a liquid coolant inlet tube and a liquid coolant outlet tube,

Predictability score

1

Milestone 4

Goal : To create a google site to for landing the USPTO application and creating a demonstration video for running the application and documentation.

Steps:

1) We will create a Google site where we can land our USPTO application.

Here is the link: https://sites.google.com/d/1A5rgwxAxZbYCsSemqmOLxPl88Lh8-LWm/p/1ajaAoktxj7Y3IlUgo3R3aSVE3Ul27ymg/edit

2) The application demonstration video is uploaded the in following github link: https://github.com/vm567/Project/blob/milestone-4/README.md

Conclusion: By issuing patents and registering trademarks, the United States Patent and Trademark Office (USPTO) encourages innovation, safeguards intellectual property, and promotes economic growth in the country.

Thus, we have implemented each milestone and created a required output.